WOnDA

Synodinos, Dionysios G.; Avgeriou, Paris

Published in:
EPRINTS-BOOK-TITLE

Publication date:
2002

WOnDA: an Extensible Multi-platform Hypermedia Design Model

Dionysios G. Synodinos[1], Paris Avgeriou[2]

[1]National Technical University of Athens, Network Management Center, Heroon Polytechniou 9, Zografou 157 80, Greece, dsin@noc.ntua.gr
[2]National Technical University of Athens, Software Engineering Laboratory, Heroon Polytechniou 9, Zografou 157 80, Greece, pavger@softlab.ntua.gr

Abstract. The design and development of hypermedia applications that are deployed on the Web and other delivery platforms is largely conducted on an intuitive, ad hoc basis, thus resulting in inefficient systems that are hard to modify, maintain and port to alternative platforms. There are now justifiable research and development efforts that attempt to formalize the engineering process of such systems in order to achieve certain quality attributes like modifiability, maintainability and portability. This paper presents such an attempt for designing a conceptual model of a hypermedia application that allows for easy update and alteration of its content as well as its presentation and also allows for deployment in various platforms. In specific this model explicitly separates the hypermedia content from its presentation to the user, by employing XML content storage and XSL transformations. Our work is based upon the empirical results of designing, developing and deploying hypermedia applications in various platforms, and on the practices of well-established hypermedia engineering techniques.

1 Introduction

During the last decade web pages have evolved to a point of becoming too complex with all the inline client scripts (e.g. JavaScript) and styles rules in order to facilitate the ever-growing and often conflicting demands for enhanced usability and impressive 'look and feel'. Therefore the daily task of updating content can no longer be performed by a novice in HTML, but instead designated professionals with a solid background on web authoring and a clear understanding of the architecture of the certain site must be utilized.

To make matters worse, even though the functional and non-functional requirements of web sites can be satisfied through the use of standard technologies like the latest versions of HTML and JavaScript, web authors find themselves using more and more proprietary solutions to facilitate customer needs and overcome the weak compatibility in this area, that major browsers offer. This situation can lead to the need of internal branching in sites that heavily deploy cutting-edge DHTML or browser specific code. Thus the number of maintained templates grows and the administrative tasks mentioned above can become overwhelming.

On top of everything else, alternative versions of web sites have become common practice nowadays to satisfy technological, political, economic and social goals. These alternative versions include mobile devices sites, accessibility sites, printable versions of sites, voice-enabled sites.
The World Wide Web Consortium reacted quite early to the above, accumulated problems with the launch of XML and the related family of technologies, in order to separate the content from the rest of the information such as presentation rules, metadata, active components etc. The question now is, given the XML technology, how can a site be engineered in order for it to achieve modifiability, maintainability and portability.
In this paper we attempt to solve the above problems by proposing an XML-based multi-tier model, named **WOnDA** (**W**rite **On**ce, **D**eliver **A**nywhere), which is established upon the separation of the actual content, active widgets, presentation rules and the page generation process. The presentation rules are themselves separated into a set of rules for transforming the actual content, the various widgets and the layout of the pages for every one of the presentational domains. Furthermore for domains that draw upon HTML (web browser versions) there is an option of defining one more presentational layer that lays above all the rest and utilizes the W3C's Cascading Style Sheet (CSS) specifications.
The proposed model is based on an XML repository that holds the actual (textual) content and utilizes the power of eXtensible Style Sheet (XSL) transformations in a hierarchical manner for providing a rich set of formats to deploy content It also facilitates effortless administration, the ability to easily add new formats and fast/clear refactoring of old ones. Also, since there is a complete separation between data and presentation, the development of content can become a streamlined process that doesn't deal with the complexity of the underlying structure of the chosen presentational domains.
The structure of the rest of this paper is as follows: Section 2 introduces a short literature review, comprised of the most significant approaches in designing hypermedia applications. Section 3 analyses the proposed model and its philosophy. Section 4 presents a complete case study of a single page that is created with the aid of the model and deployed in three platforms. Finally section 5 wraps up with ideas for future work.

2 Literature Review

During the past years there have been several attempts to formalize processes, models, methods, techniques and best practices for developing hypermedia applications [1].
The **Dexter Hypertext Reference Model** [2] was perhaps the first successful attempt at modeling hypermedia applications and has been the predecessor and source of inspiration for many of the approaches that followed. Another widely used approach of hypermedia design modeling is **HDM** [3] and its successor HDM2 [4], where emphasis is given in specifying a semantic schema of the application before it is developed. Its main feature is the definition of a number of dimensions, along which a

hypermedia application can be modeled: structure, navigation, behavior, user control and presentation. The **Object-Oriented Hypermedia Design Model (OOHDM)** [5] is based on the separation of the hypermedia application's process into four phases: Conceptual Design, Navigation Design, Abstract Interface Design and Implementation. A CASE Tool named OOHDM-Web [6], allows implementation of hypermedia applications as CGI scripts that produce dynamically generated pages, whose contents are fed from a database and integrated with predefined templates. Another methodology that provides step-by-step hypermedia development is **Relationship Management methodology** (RMM) [7]. RMM offers complete representation of the semantic schema and the navigational schema, follows the traditional E-R model to standardize the conceptual and navigational design but gives limited support to the interface design.
The **CC/PP framework** [8] of the World Wide Web Consortium aims to provide a common way for clients to express their capabilities and preferences to the server that originates content. WOnDA can be intergraded with the CC/PP model in the manner that different presentation formats can be created in an asynchronous way to satisfy the device profiles collected from a central or various distributed profile registries. But since WOnDA proposes that, creation of hypermedia context is done prior to user request (static content), a scenario where the content is adapted dynamically to the capabilities of the device by reading the device's profile in real time, is not possible. The later though can be helpful in a database-driven website. Also an important approach is **Sisl** [9], which is an architecture and domain-specific language for designing and implementing services with multiple user interfaces. It aims to the decoupling of interface from service logic, by employing an event-based model of services that allows service providers to support interchangeable user interfaces to a single source of service logic or data.

3 The WOnDA Model

A macroscopic view of the model 's functionality is illustrated in Figure 1. The content is authored in an editor that could be a simple text editor or any contemporary word processor like MS Word or Word Perfect. It may include text and references to other media such as images, video, sound, animation etc. The content then is translated into an XML file that properly describes its various parts like text, hyperlinks, video clips etc. In sequence, XSL

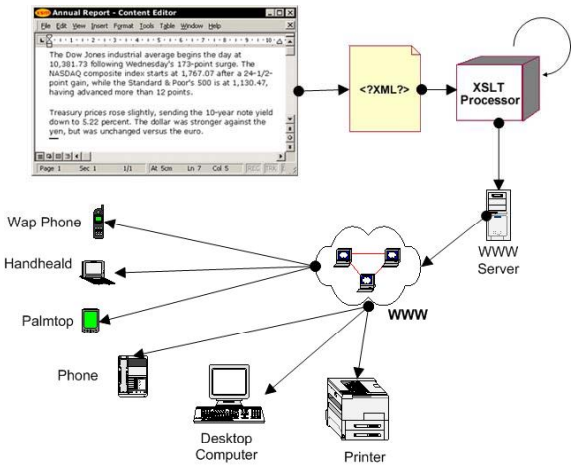

Figure 1 A macroscopic view of the model

transformations are utilized to impose style rules and presentation layout, add active objects, and generate the page in its final form, e.g. HTML page, WML page etc. The final page is then served to the appropriate client through the Internet by being published to a Web Server.

The detail that is missing from the above figure, is the mechanism that deals with the XML files and XSL transformations that translate raw content into a specific delivery platform. This mechanism is described in the rest of this section and aims to serve as a guide for the construction of maintainable, modifiable, and portable hypermedia applications.

In order to describe the model we utilize the Unified Modeling Language [10] (http://www.rational.com/uml), a widely adopted modeling language in the software industry and an Object Management Group standard [http://www.omg.org/]. Furthermore in order to define the model we have designed a UML meta-model that helps define what is a well-formed model – i.e. one that is syntactically correct.

The model described here considers only static hypermedia pages and every page is comprised of the following elements:

1. The actual content of the page
2. A set of navigational or promotional active objects or widgets like navigation bars, search boxes, menus, logos, ads, banners etc.
3. The general layout of the page meaning the positioning of all the above in the browser window and the rest of the markup envelope that is needed in order for the page to be syntactically valid.
4. Hyperlinks to other hypermedia pages

It is noted that this is a simplified and superficial model of a hypermedia page because the aim of the model is not to model hypermedia applications in general but merely to separate content from the rest of the information and generate multiple versions of hypermedia applications. In other words the proposed model is considered to be in a lower abstraction layer than usual hypermedia design models such as HDM [3], OODHM [5], RMM [7], WebML [11] etc.

The principles of the proposed meta-model are the following:

1. The actual content of each hypermedia page is kept in one XML file. These files will be referred to as **Content Pages** (CPs). CPs represent published pages as abstract data entities without taking into account any presentation aspects derived by the desired formats.
2. The task of providing content rendering information is left up to a set of XSL files, which will be referred to as **Content Transformers** (CTs). If we define a set of N versions for the site under construction, every version is exactly identical to all the others in terms of textual information since this information is provided by the CPs, but the versions differ in the layout, functionality, style and the markup that they're written in. For every one of these versions we define a CT which describes the rules necessary to transform the content provided by the respectable CPs.
3. In the fashion of CPs and CTs for the textual content we define **Widget Pages** (WPs) and **Widget Transformers** (WTs), which hold the necessary data and presentation rules respectively for the widgets used.
4. Metadata that are specific to the content page, as in the HTML <META> element, used throughout a version or even throughout the entire site are kept in an XML file, which will be referred to as **Content Page Metadata** (CPM).

5. Metadata that are specific to a version, e.g. character encoding information, are kept in another XML file, called **Version-Specific Metadata Page (VSMP).**
6. Both content page-specific and version-specific metadata are rendered by another transformer called **Metadata Transformer (MT).**
7. For every one of the different versions we define an XSL file, which describes the rules necessary to generate the page layout that is restricted in the context of the version. These XSL files, which will be referred to as **Version Builders (VBs)**, do not contain any information about the rules we need to render the textual content drawn from the CPs nor the widgets used. They rather define the general layout of the page meaning the positioning of all the above in the browser window and the rest of the markup envelope that is needed in order for the page to be syntactically valid for the corresponding presentational domain. The information about client scripts and CSS, are referenced by the VB, where needed and are imported in the document e.g. via the HTML <LINK> element.

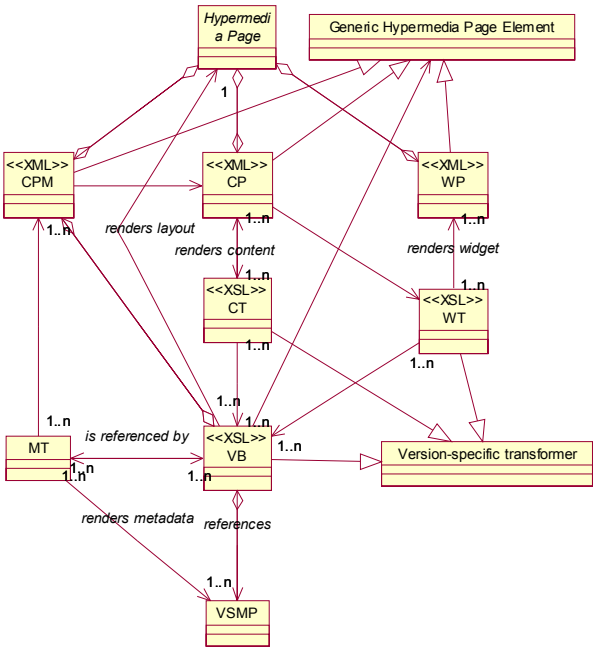

Figure 2 – the hypermedia page elements and the transformers

Figure 2 depicts the relationship between the above model elements. Content Pages, Widget Pages and Content Page Metadata are XML files and are all specializations of the class "Generic Hypermedia Page Element". They are also connected with an aggregation relationship with the "Hypermedia Page" class, which means that they are all part of a hypermedia page. Content Transformer and Widget Transformer are XSL files that render the corresponding Content Pages and Widget Pages. Furthermore it is obvious that Content Metadata Pages are related to Content Pages, in the sense that metadata describe the content. Moreover, Content Metadata Pages and Version-Specific Metadata Page are rendered by the Metadata Transformer. Finally Version Builders use all the other transformers to render the layout of the hypermedia page and insert the appropriately transformed content, widgets and metadata into them. Content Transformers, Widget Transformers and the Version Builder are specializations of the "Version-Specific Transformer" class.

8. For every site there is a main registry that holds information used by the main

processing mechanism. This master registry is the **Site Index Page (SIP)**, which holds all the file system (or network) paths to the CPs and links every one of them to a registry specific for it, called **Page Registry (PR)**. PRs link together CPs and PTs for the various versions.

9. All the above are parsed by a processing shell, which will be referred to as ***Site Builder*** and provides the web site administrator with a web interface to generate or update certain pages, entire versions of the site etc.

Figure 3 depicts the page creation process that takes place with the aid of the last three elements. The Page Registry gathers all the necessary data from the Content Page, the Content Transformer, the Content Page Metadata and the Version Builder. The Site Builder parses the Site Index Pages to look for all the Page Registries, performs the transformations and generates the hypermedia page of the appropriate format, as illustrated in Figure 4 through a UML activity diagram.

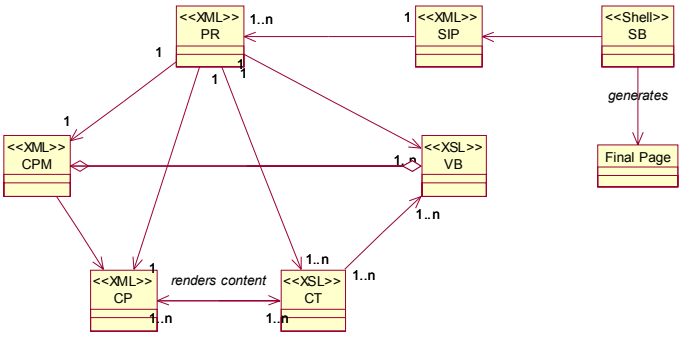

Figure 3 – The Page Generation Process

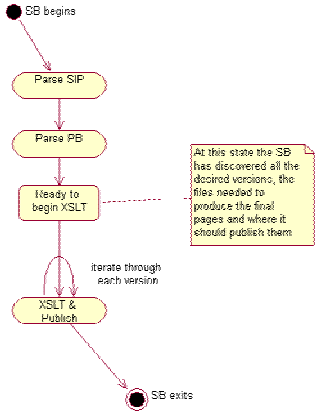

Figure 4 - The functionality of the SB through a UML activity diagram.

4 A Case Study

In order to explicate WOnDA, we'll examine in detail the design of a web page of a laboratory that offers references to external web sites holding tutorials on XML. This page should be accessible from the Mozilla browser, from a normal WAP phone

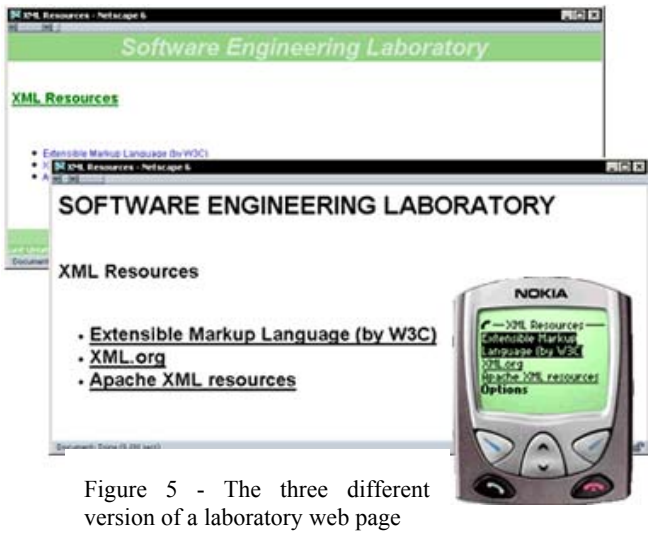

Figure 5 - The three different version of a laboratory web page

and we must also provide it in an easily readable format for students with disabilities. Figure 5 depicts the final product in the desired formats. Figure 6 depicts the implementation of the meta-model described earlier in the case of the "XML Resources" page. This is an instance of the meta-model, i.e. a model per se that is consisted of instance of all the necessary meta-model elements. It is noted that since this example is rather simple, not all the meta-model elements are needed to describe it.

Figure 6

1 Content Pages (CPs)

As mentioned earlier the actual content of a page is described inside a CP. The following code listing holds both the textual information of our "XML Resources" page and the needed references to external resources like the external links. The XML vocabulary used here is pretty self-explanatory since it uses widely recognized terms like "paragraph", "list" etc. for the names of the elements used.

Example CP for the "XML Resources" page

```
<data>
<title value="XML Resources"/>
<content>
<paragraph>
  <emphasize>XML Resources</emphasize>
</paragraph>
<list type="unordered">
  <ListElement>
    <link location="http://www.w3.org/xml">
      Extensible Markup Language (by W3C)
    </link>
  </ListElement>
  <ListElement>
    <link location="http://www.xml.org">
      XML.org
    </link>
  </ListElement>
  <ListElement>
    <link location="http://xml.apache.org">
      Apache XML resources
    </link>
  </ListElement>
</content>
</data>
```

2 Content Transformers (CT)

The CTs for our page will provide the processing shell (SB) the necessary rules in order to transform the XML code of the CP. The final form of such a CT depends on the elements used inside the CP. In the following code listing we suggest a set of four XSL rules to handle the transformation of the elements

Example CT for the "XML Resources" page

```
<xsl:template match="paragraph">
  <p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template
match="list[@type='unordered']">
  <ul><xsl:apply-templates/></ul>
</xsl:template>
<xsl:template
```

<paragraph>, <list> and <link>. These transformations will produce an HTML fragment that is suitable for the Mozilla browser version of the page.

```
match="list[@type='unordered']/ListEle
ment">
  <li class="UnorderedListElement">
<xsl:apply-templates/></li>
</xsl:template>

<xsl:template match="link">
  <a href="{@location}"><xsl:apply-
templates/></a>
</xsl:template>
```

3 Widget Pages (WP) and Widget Transformers (WT)

The paradigm of CPs and CTs is also used for the WPs and the WTs as the code listing on the right shows. This is an example of XSL transformations for the WP for the "XML Resources" footer as described above, for the Mozilla version.

```
<widget name="Footer">
  <element type="TextLink"
    label="Homepage" link="CP.Home.xml"/>
  <element type="TextLink"
    label="Contact" link="CP.Contact.xml"/>
  <element type="TextLink"
    label="Search" link="CP.Search.xml"/>
  <element type="TextLink"
    label="Webmaster"
link="mailto:webmaster@lab"/>
</widget>

<xsl:output encoding='ISO-8859-1'/>
<xsl:template match="widget[@name='Footer']">
  <p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="element">
   <a href="{@link}">
    <xsl:value-of select=".[@label]"/>
  </a>  
</xsl:template>
</xsl:stylesheet>
```

4 Metadata Pages (MP)

For the purpose of supplying our page with the required meta-data we use both the CPM and the VSMP mechanism described earlier for the model. In the following code sample we describe a way to handle three trivial cases of meta data information that appears in web pages.

Example nodes for Metadata pages either CPM or VSMP

```
<MetaDataElement type="author"
  content="Paris Avgeriou"/>

<MetaDataElement type="http-equiv"
  value="Expires"
  content="Tue,  20  Aug  2002  14:25:27
GMT"/>
```

5 Version Builders (VBs)

The VB of every version will import the XSLT rules described in the CT, WT and MT of the later version and

The skeleton of the VB for the Mozilla browser

```
<xsl:import href="CT.Web Mozilla.xsl"/>
<xsl:import
href="WT Footer Web Mozilla.xsl"/>
```

provide the overall envelope to bind rules and content. The following code sample is a simple VB for the Mozzilla browser.

Key elements of the XSL sample on the right are the imports of the other XSL rules, in the <xsl:import> tag, and the use of the document() function to include any other XML document besides the CP.

```
href="WT.Footer.Web Mozilla.xsl"/>
<xsl:import
href="WT.Footer.Web Mozilla.xsl"/>
<xsl:variable name="Footer"
  select="document('WP.Footer.xml')"/>
<xsl:variable name="CPM"

select="document('CPM.XMLResources.xml')"/>
<xsl:variable name="VSMP"

select="document('VSMP.Web_Mozilla.xml')"/>
<xsl:output encoding='ISO-8859-1'/>
  <xsl:template match="/">
  …
  </xsl:template>
   </xsl:stylesheet>
```

6 Site Index Page (SIP) and the Page Registry (PR)

In the following part we give an example of a SIP and a PR. The syntax is again self-explanatory.

```
<pages>
  <page description="foo"
    PR_Path="PR.foo.xml"/>
  <page description="bar"
    PR Path="PBR.bar.xml"/>
  <page description="XML Resources"
    PR Path="PBR.XmlResources.xml"/>
</pages>

  <ContentPage path="CP.XmlResources.xml"/>
  <version name="Web Mozilla"
    CT Path="CT.Web Mozilla.xsl"
    Pub Path="/www root/html/XMLResources.html"/>
  <version name="WAP"
    CT_Path="CT.WAP.xsl"
    Pub Path="/www root/wml/XMLResources.wml />
  <version name="Accesibility"
    CT Path="CT.Accesibility.xsl"
    Pub_Path="/www_root/accs/XMLResources.html />
```

7 Processing Shell (Site Builder - SB)

As mentioned above the entity responsible for applying the final XSL Transformation and publishing the pages is the Site Builder. This module was implemented as a Java 2 Application that consists of a web-based front-end for the administrator and an internal mechanism to apply XSLT using any Java-XML package.

5 Future Work

Although the principles described above provide a solid foundation, they do not deal with the entirety of the diverse scenarios that are popular in contemporary hypermedia applications. To better handle this without affecting the simplicity of WOnDA, which is one of its strongest attributes, we plan to extend this model in a modular way having different *flavors* of the original approach, such as:

- **Distributed flavor**: In cases of web sites of large organizations or multinational corporations, the operations of content authoring, style development, page generation and publishing can be executed in a distributed way, over a plethora of locations worldwide, in a pattern designated by the business plan or the established workflow of the respective organization or corporation.
- **Multilingual flavor**: multilingual context should be better handled.
- **Database & server side scripting flavor:** the model should deal specifically with situations where the developer needs to systematically draw information from a database or merely execute HTML-embedded server side scripts.
- **User personalization flavor:** the model should take into account the practice of dynamically adapting pages according to the site visitors.

Finally, an interesting issue that is under research is the way that WOnDA can benefit from all the work recently done in the database area in order to deal with XML documents. This is caused by the variety of proposed models for storing and accessing XML information both natively and on object-relational databases. To examine this issue more closely the authors are actively involved with "INEX: Initiative for the Evaluation of XML retrieval" [http://qmir.dcs.qmul.ac.uk/INEX/].

References

1. Lowe, W. Hall, Hypermedia & the Web, an Engineering Approach, John Wiley & Sons: 1999.
2. F. Halasz and M. Schwartz. "The Dexter Hypertext Reference Model". *Communications of the ACM*, 37(2):30-39,Feb.1994
3. F. Garzotto, D. Schwabe, P. Paolini, "HDM- A Model Based Approach to Hypermedia Application Design", ACM Transactions on Information Systems, Vol. 11, #1, Jan. 1993, pp. 1-26.
4. F. Garzotto, L. Mainetti, P. Paolini, "Navigation in Hypermedia Applications: Modeling and Semantics", Journal of Organizational Computing and Electronic Commerce. 6, 3 (1996), 211-238.T.
5. D. Scwhabe & G. Rossi, "The Object-Oriented Hypermedia Design Model (OOHDM)", Communications of the ACM, Vol. 35, no. 8, August 1995.
6. D. Schwabe & R. de Almeida Pontes, "OOHDM-WEB: Rapid Prototyping of Hypermedia Applications in the WWW", Tech. Report MCC 08/98, Dept. of Informatics, PUC-Rio, 1998.
7. T. Isakowitz; E. Stohr; P. Balasubramaniam, "RMM, A methodology for structured hypermedia design", *Communications of the ACM*, August 1995, pp 34-48
8. World Wide Web Consortium (W3C), "Composite Capabilities/Preference Profiles: Requirements and Architecture", W3C Working Draft 28 February, 2000.
9. T. Ballz, C. Colby, P. Danielseny, L. Jategaonkar Jagadeesany, R. Jagadeesan, K. L'aufer, P. Matagay, K. Rehory, "Sisl: Several Interfaces, Single Logic", Journal of Speech Technology, Kluwer Academic Publishers, 2000.
10. G. Booch, J. Rumbaugh, and I. Jacobson, The UML User Guide, Addison-Wesley, 1999.
11. S. Ceri. P. Fraternali, A. Bongio, "Web Modeling Language: a modelling language for designing Web sites", WWW9 Conference, Amsterdam, 5/2000.